

Project 2: Expense Tracker

“I’m a small business owner and struggle too keep tabs on my expenses, Is there a tool you can develop that Helps me manage my spending more effectively?”

Features Requested:

Expense Logging: “I need a way to easily input my expenses, categorize them, and see a history of my Spending.”

Budget Setting: “Can the software allow me to set budgets for different expense categories and alert me if I’m overspending?”

Visualization of Expenses: “it would be helpful to see graphs or charts that visually represent my sending Patterns.”

Reminder System: “Can the tool remind me when certain bills are due or when I’m approaching my budget Limits?”

Solution: 2

```
package training.DataFlair;

import java.awt.EventQueue;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import org.sqlite.SQLiteDataSource;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JComboBox;

import javax.swing.table.DefaultTableModel;

import javax.swing.JTextField;
```

```

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class ExpenseTracker {

    private static Connection conn;

    private static SQLiteDataSource ds;

    private JFrame frame;

    private JTable table;

    private JTextField dateField;

    private JTextField descField;

    private JTextField amountField;

    private JTextField nameField;

    private int currentAccountId = 0;

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            public void run() {

                try {

                    ExpenseTracker window = new ExpenseTracker();

                    window.frame.setVisible(true);

                } catch (Exception e) {

                    e.printStackTrace();

                }

            }

        });

    }

    public ExpenseTracker() {

        initDB();

        initialize();

    }

    private void initDB() {

        ds = new SQLiteDataSource();

```

```

try {
    ds = new SQLiteDataSource();
    ds.setUrl("jdbc:sqlite:ExpensesDB.db");
} catch (Exception e) {
    e.printStackTrace();
    System.exit(0);
}

try {
    conn = ds.getConnection();

    Statement statement = conn.createStatement();
    statement.executeUpdate("CREATE TABLE IF NOT EXISTS accounts (\n"
        + " id INTEGER PRIMARY KEY AUTOINCREMENT,\n"
        + " name TEXT\n"
        + ");\n"
        + "CREATE TABLE IF NOT EXISTS expenses (\n"
        + " id INTEGER PRIMARY KEY AUTOINCREMENT,\n"
        + " account_id INTEGER,\n"
        + " date TEXT,\n"
        + " description TEXT,\n"
        + " amount REAL,\n"
        + " FOREIGN KEY (account_id) REFERENCES accounts(id)\n"
        + ");\n");

    // Closing statement and connection
    statement.close();
    conn.close();

} catch (SQLException e) {
    e.printStackTrace();
    System.exit(0);
} finally {

```

```

try {
    if (conn != null) {
        conn.close();
    }
} catch (SQLException e) {
    System.err.println(e);
}
}
}

```

// Method to add account to the database

```

private void addAccount(String accountName) {
    try {
        // Open connection to the database
        conn = ds.getConnection();

        // Prepare SQL statement for inserting data into the accounts table
        String sql = "INSERT INTO accounts (name) VALUES (?)";
        PreparedStatement stmt = conn.prepareStatement(sql);

        // Set parameter for the statement
        stmt.setString(1, accountName);

        // Execute the statement to insert the data into the table
        stmt.executeUpdate();

        // Close the statement and connection
        stmt.close();
        conn.close();

        JOptionPane.showMessageDialog(frame, "Account Added Successfully");
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(frame, "Error adding account: " + e.getMessage());
    }
}
}

```

// Method to Load Data from the database into the table

```

public void loadData(DefaultTableModel model, int acId) throws SQLException {

    model.setRowCount(0);

    conn = ds.getConnection();

    String sql = "SELECT date,description,amount FROM expenses WHERE account_id = ? ";

    PreparedStatement ps = conn.prepareStatement(sql);

    ps.setInt(1, acId);

    ResultSet rs = ps.executeQuery();

    Object[] row = new Object[3];

    while (rs.next()) {

        for (int i = 0; i < row.length; i++) {

            row[i] = rs.getObject(i + 1);

        }

        model.addRow(row);

    }

    ps.close();

    conn.close();

}

```

// method to get account details such as account name and total expense from the

// database

```

private String[] getAccountDetails(int accountId) {

    double totalExpense = 0.0;

    String accountName = null;

    try {

        conn = ds.getConnection();

        // Prepare SQL statement to retrieve account name

        String accountSql = "SELECT name FROM accounts WHERE id = ?";

        PreparedStatement accountStmt = conn.prepareStatement(accountSql);

        accountStmt.setInt(1, accountId);

        // Execute the account statement and retrieve the result

        ResultSet accountResult = accountStmt.executeQuery();

        if (accountResult.next()) {

```

```

        accountName = accountResult.getString("name");

        // Prepare SQL statement to calculate total expense amount
        String expenseSql = "SELECT SUM(amount) AS total FROM expenses WHERE account_id = ?";
        PreparedStatement expenseStmt = conn.prepareStatement(expenseSql);
        expenseStmt.setInt(1, accountId);

        // Execute the expense statement and retrieve the result
        ResultSet expenseResult = expenseStmt.executeQuery();
        if (expenseResult.next()) {
            totalExpense = expenseResult.getDouble("total");
        }
        expenseResult.close();
        expenseStmt.close();
    }
    accountResult.close();
    accountStmt.close();
    conn.close();
} catch (SQLException e) {
    JOptionPane.showMessageDialog(frame, "Error retrieving account details: " + e.getMessage());
}

String[] detail = { accountName, totalExpense + "" };
return detail;
}

```

// Method to add the expense into the currently selected account

```

private void addExpense(int accountId, String date, String description, double amount) {
    try {
        // Open connection to the database
        conn = ds.getConnection();

        // Prepare SQL statement for inserting data into the expenses table
        String sql = "INSERT INTO expenses (account_id, date, description, amount) VALUES (?, ?, ?, ?)";
        PreparedStatement stmt = conn.prepareStatement(sql);

        // Set parameters for the statement
    }
}

```

```
stmt.setInt(1, accountId);
stmt.setString(2, date);
stmt.setString(3, description);
stmt.setDouble(4, amount);
// Execute the statement to insert the data into the table
stmt.executeUpdate();
// Close the statement and connection
stmt.close();
conn.close();
```

```
    JOptionPane.showMessageDialog(frame, "Expense added successfully to Account ID: " + accountId);
} catch (SQLException e) {
    JOptionPane.showMessageDialog(frame, "Error adding expense: " + e.getMessage());
}
}
```

```
// Method to update the comboBox with data from the database
public void updateCombox(JComboBox<String> cbx) throws SQLException {
    cbx.removeAll();
    conn = ds.getConnection();
    String sql = "SELECT * FROM accounts;";
    PreparedStatement ps = conn.prepareStatement(sql);
    ResultSet rs = ps.executeQuery();

    while (rs.next()) {
        cbx.addItem(rs.getString("id") + " | " + rs.getString("name"));
    }
    rs.close();
    ps.close();
    conn.close();
}
```

```
/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    frame = new JFrame();
    frame.setBounds(100, 100, 600, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    frame.setTitle("Expense Tracker by DataFlair");

    JPanel toppanel = new JPanel();
    toppanel.setBounds(0, 0, 590, 58);
    frame.getContentPane().add(toppanel);
    toppanel.setLayout(null);

    JLabel lblSelectAc = new JLabel("Select A/C:");
    lblSelectAc.setBounds(0, 0, 75, 15);
    toppanel.add(lblSelectAc);

    JComboBox accBox = new JComboBox();
    accBox.setBounds(86, 0, 130, 24);
    toppanel.add(accBox);

    JLabel lblName = new JLabel("Name:");
    lblName.setBounds(10, 37, 70, 15);
    toppanel.add(lblName);

    nameField = new JTextField();
    nameField.setColumns(10);
    nameField.setBounds(86, 27, 130, 30);
    toppanel.add(nameField);
```



```

JButton btnAddAc = new JButton("Add A/C");
btnAddAc.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        addAccount(nameField.getText());
        try {
            updateCombox(accBox);
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
    }
});
btnAddAc.setBounds(223, 27, 117, 30);
toppanel.add(btnAddAc);

```

```

JButton btnSelect = new JButton("Select");
btnSelect.setBounds(223, 0, 117, 25);
toppanel.add(btnSelect);

```

```

JScrollPane scrollPane = new JScrollPane();
scrollPane.setEnabled(false);
scrollPane.setBounds(0, 60, 590, 211);
frame.getContentPane().add(scrollPane);

```

```

table = new JTable();
table.setBounds(0, 0, 0, 0);
table.setModel(new DefaultTableModel(
    new Object[][] {
        { null, null, null },
    },
    new String[] {
        "Date", "Description", "Amount"
    }
));

```

```
scrollPane.setViewportView(table);
```

```
JPanel bottomPanel = new JPanel();
```

```
bottomPanel.setBounds(0, 270, 600, 90);
```

```
frame.getContentPane().add(bottomPanel);
```

```
bottomPanel.setLayout(null);
```

```
JLabel dateLabel = new JLabel("Date:");
```

```
dateLabel.setBounds(0, 5, 50, 15);
```

```
bottomPanel.add(dateLabel);
```

```
dateField = new JTextField();
```

```
dateField.setBounds(50, 5, 114, 30);
```

```
bottomPanel.add(dateField);
```

```
dateField.setColumns(10);
```

```
JLabel descLabel = new JLabel("Description:");
```

```
descLabel.setBounds(180, 5, 90, 15);
```

```
bottomPanel.add(descLabel);
```

```
descField = new JTextField();
```

```
descField.setColumns(10);
```

```
descField.setBounds(270, 5, 114, 30);
```

```
bottomPanel.add(descField);
```

```
JLabel amountLabel = new JLabel("Amount:");
```

```
amountLabel.setBounds(390, 5, 70, 15);
```

```
bottomPanel.add(amountLabel);
```

```
amountField = new JTextField();
```

```
amountField.setColumns(10);
```

```
amountField.setBounds(456, 5, 114, 30);
```

```
bottomPanel.add(amountField);
```

```
JButton btnAddExpense = new JButton("Add");
```

```
btnAddExpense.addActionListener(new ActionListener() {
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        addExpense(currentAccountId, dateField.getText(), descField.getText(),
```

```
            Double.valueOf(amountField.getText()));
```

```
        try {
```

```
            loadData((DefaultTableModel) table.getModel(), currentAccountId);
```

```
        } catch (SQLException e1) {
```

```
            e1.printStackTrace();
```

```
        }
```

```
    }
```

```
});
```

```
btnAddExpense.setBounds(239, 39, 117, 25);
```

```
bottomPanel.add(btnAddExpense);
```

```
JLabel lblTotalExpense = new JLabel("Total Expense : ");
```

```
lblTotalExpense.setBounds(22, 75, 120, 15);
```

```
bottomPanel.add(lblTotalExpense);
```

```
JLabel lbltotalAmount = new JLabel("--");
```

```
lbltotalAmount.setBounds(143, 75, 70, 15);
```

```
bottomPanel.add(lbltotalAmount);
```

```
JLabel lblCurrAcc = new JLabel("Current Acc Name:");
```

```
lblCurrAcc.setBounds(270, 76, 130, 15);
```

```
bottomPanel.add(lblCurrAcc);
```

```
JLabel lblAccountName = new JLabel("Account Name");
```

```
lblAccountName.setBounds(412, 76, 130, 15);
```

```
bottomPanel.add(lblAccountName);
```

```
btnSelect.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        String accountId = (String) accBox.getSelectedItem();  
        accountId = accountId.substring(0, accountId.indexOf('|'));  
        currentAccountId = Integer.valueOf(accountId);  
        try {  
            loadData((DefaultTableModel) table.getModel(), currentAccountId);  
        } catch (NumberFormatException e1) {  
            e1.printStackTrace();  
        } catch (SQLException e1) {  
            e1.printStackTrace();  
        }  
        String details[] = getAccountDetails(currentAccountId);  
        lbltotalAmount.setText(details[1]);  
        lblAccountName.setText(details[0]);  
    }  
});
```

```
try {  
    updateCombox(accBox);  
} catch (SQLException e1) {  
    e1.printStackTrace();  
}  
}
```

```
}
```

Output 2:

Expense Tracker by DataFlair

Select A/C: 3|Travel

Select

Name:

Add A/C

Date	Description	Amount
2023-04-04	Flight Ticket	3500.0
2023-04-05	Hotel Stay	2000.0

Date:

Description:

Amount:

Add

Total Expense : 5500.0

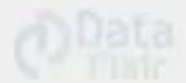
Current Acc Name: Travel

Expense Tracker by DataFlair



Select A/C: 3|Travel

Select



Name:

Add A/C

Date	Description	Amount
2023-04-04	Flight Ticket	3500.0
2023-04-05	Hotel Stay	2000.0

Message



Expense added successfully to Account ID: 3

OK



Date:

2023-04-04

Description:

Train Ticket

Amount:

2000

Add



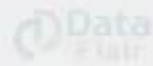
Total Expense : 5500.0

Current Acc Name: Travel

Expense Tracker by DataFlair

Select A/C: 1|Personal

Select



Name:

Add A/C

Date	Description	Amount
2023-04-01	Groceries	1500.0
2023-04-02	Restaurant	500.0
2023-04-05	Restaurant	300.0



Date:

Description:

Amount:



Add

Total Expense : 2300.0

Current Acc Name: Personal

Project 1 : Inventory Management System

"I run small retail store and struggle to keep track of my inventory. Can you create a software solution that Helps me manage my products, sales, and inventory more efficiently?"

Features Requested:

Product Management: "I need a system where I can easily add new products, update their details, and Remove outdated ones."

Stock Tracking: "Can the software track how many of each item I have in stock and notify me when I need to Restock?"

Sales Recording: "I want to record each sale and have it automatically update my inventory levels."

Report Generation: "it would be great to generate reports showing what's selling well and what's low in Stock."

Solution 1:

➤ MySQL setup

```
CREATE DATABASE `inventory` ;
```

```
USE `inventory`;
```

```
CREATE TABLE `currentstock` (  
  `productcode` varchar(45),  
  `quantity` int,  
  PRIMARY KEY (`productcode`)  
);
```

```
INSERT INTO `currentstock` VALUES  
('prod1',146),('prod2',100),('prod3',202),('prod4',172),('prod5',500),('prod6',500),('prod7',10),('prod8',20);
```

```
CREATE TABLE `customers` (  
  `cid` int NOT NULL AUTO_INCREMENT,  
  `customercode` varchar(45),  
  `fullname` varchar(45),  
  `location` varchar(45),  
  `phone` varchar(45),
```



```
PRIMARY KEY (`cid`)
```

```
);
```

```
INSERT INTO `customers` VALUES (301,'vip1','John Seed','New York','9818562354'),(302,'vip2','Jacob Seed','Texas','9650245489'),(303,'std1','Ajay Kumar','Mumbai','9236215622'),(304,'std2','Astha Walia','Chandigarh','8854612478'),(306,'vip3','Madhu Chitkara','Chandigarh','9826546182');
```

```
CREATE TABLE `products` (
```

```
`pid` int NOT NULL AUTO_INCREMENT,
```

```
`productcode` varchar(45),
```

```
`productname` varchar(45),
```

```
`costprice` double,
```

```
`sellprice` double,
```

```
`brand` varchar(45),
```

```
PRIMARY KEY (`pid`),
```

```
UNIQUE KEY `productcode_UNIQUE` (`productcode`)
```

```
);
```

```
INSERT INTO `products` VALUES
```

```
(111,'prod1','Laptop',85000,90000,'Dell'),(112,'prod2','Laptop',70000,72000,'HP'),(113,'prod3','Mobile',60000,64000,'Apple'),(114,'prod4','Mobile',50000,51000,'Samsung'),(121,'prod5','Charger',2000,2100,'Apple'),(122,'prod6','Mouse',1700,1900,'Dell'),(128,'prod7','Power Adapter',3000,3500,'Dell'),(129,'prod8','Smart Watch',15000,17000,'Apple');
```

```
CREATE TABLE `purchaseinfo` (
```

```
`purchaseID` int NOT NULL AUTO_INCREMENT,
```

```
`suppliercode` varchar(45),
```

```
`productcode` varchar(45),
```

```
`date` varchar(45),
```

```
`quantity` int,
```

```
`totalcost` double,
```

```
PRIMARY KEY (`purchaseID`)
```

```
);
```

```
INSERT INTO `purchaseinfo` VALUES (1001,'sup1','prod1','Wed Jan 14 00:15:19 IST
2021',10,850000),(1002,'sup1','prod6','Wed Jan 14 00:15:19 IST 2021',20,34000),(1003,'sup2','prod3','Wed Jan 14
00:15:19 IST 2021',5,300000),(1004,'sup2','prod5','Wed Jan 14 00:15:19 IST 2021',5,10000),(1005,'sup3','prod2','Wed Jan
14 00:15:19 IST 2021',2,140000),(1006,'sup4','prod4','Wed Jan 14 00:15:19 IST
2021',2,100000),(1009,'sup2','prod3','Wed Sep 01 04:11:13 IST 2021',2,120000),(1010,'sup1','prod7','Wed Sep 01
04:25:06 IST 2021',10,30000),(1011,'sup2','prod8','Fri Sep 03 00:00:00 IST 2021',20,300000);
```

```
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45),
  `location` varchar(45),
  `phone` varchar(10),
  `username` varchar(20),
  `password` varchar(200),
  `usertype` varchar(45),
  PRIMARY KEY (`id`)
);
```

Code for IMS:

- ConnectionFactory.java for database connection

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
public class ConnectionFactory {

    Connection conn = null;
    Statement statement = null;
    ResultSet resultSet = null;

    public ConnectionFactory(){
        try {
```

```

        Class.forName("com.mysql.cj.jdbc.Driver");

        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/inventory", "root", "root");

        statement = conn.createStatement();

    } catch (Exception e) {

    }

}

```

```

public Connection getConn() {

    try {

        Class.forName("com.mysql.cj.jdbc.Driver");

        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/inventory", "root", "root");

        System.out.println("Connected successfully.");

    } catch (Exception e) {

    }

    return conn;

}

```

//Login verification method

```

public boolean checkLogin(String username, String password, String userType){

    String query = "SELECT * FROM users WHERE username="

        + username

        + " AND password="

        + password

        + " AND usertype="

        + userType

        + " LIMIT 1";

    try {

        resultSet = statement.executeQuery(query);

        if(resultSet.next()) return true;

    } catch (Exception ex) {

    }

}

```

```
        return false;
    }

}
```

➤ CustomerDAO

```
import javax.swing.*.*;
import java.sql.*;
import javax.swing.table.DefaultTableModel;
import java.util.Locale;
import java.util.Vector;

class CustomerDTO {

    int custID;
    String custCode, fullName, location, phone;
    double debit, credit, balance;

    public int getCustID() {
        return custID;
    }

    public void setCustID(int custID) {
        this.custID = custID;
    }

    public String getCustCode() {
        return custCode;
    }

    public void setCustCode(String custCode) {
        this.custCode = custCode;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }

    public String getLocation() {
        return location;
    }
}
```

```

    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public double getDebit() {
        return debit;
    }

    public void setDebit(double debit) {
        this.debit = debit;
    }

    public double getCredit() {
        return credit;
    }

    public void setCredit(double credit) {
        this.credit = credit;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }
}

```

```

public class CustomerDAO {
    Connection conn = null;
    PreparedStatement prepStatement= null;
    Statement statement = null;
    ResultSet resultSet = null;

    public CustomerDAO() {
        try {

```

```

        conn = new ConnectionFactory().getConn();
        statement = conn.createStatement();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

// Methods to add new custoemr

```

public void addCustomerDAO(CustomerDTO customerDTO) {
    try {
        String query = "SELECT * FROM customers WHERE fullname="
            +customerDTO.getFullName()
            + "" AND location="
            +customerDTO.getLocation()
            + "" AND phone="
            +customerDTO.getPhone()
            + "";
        resultSet = statement.executeQuery(query);
        if (resultSet.next())
            JOptionPane.showMessageDialog(null, "Customer already exists.");
        else
            addFunction(customerDTO);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public void addFunction(CustomerDTO customerDTO) {
    try {
        String query = "INSERT INTO customers VALUES(null,?,?,?,?)";
        preparedStatement = conn.prepareStatement(query);
        preparedStatement.setString(1, customerDTO.getCustCode());
        preparedStatement.setString(2, customerDTO.getFullName());
        preparedStatement.setString(3, customerDTO.getLocation());
        preparedStatement.setString(4, customerDTO.getPhone());
        preparedStatement.executeUpdate();
        JOptionPane.showMessageDialog(null, "New customer has been added.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

// Method to edit existing customer details

```

public void editCustomerDAO(CustomerDTO customerDTO) {
    try {
        String query = "UPDATE customers SET fullname=?,location=?,phone=? WHERE customercode=?";
        preparedStatement = conn.prepareStatement(query);
        preparedStatement.setString(1, customerDTO.getFullName());
        preparedStatement.setString(2, customerDTO.getLocation());
    }
}

```

```

        preparedStatement.setString(3, customerDTO.getPhone());
        preparedStatement.setString(4, customerDTO.getCustCode());
        preparedStatement.executeUpdate();
        JOptionPane.showMessageDialog(null, "Customer details have been updated.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

// Method to delete existing customer

```

public void deleteCustomerDAO(String custCode) {
    try {
        String query = "DELETE FROM customers WHERE customercode='" + custCode + "'";
        statement.executeUpdate(query);
        JOptionPane.showMessageDialog(null, "Customer removed.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

// Method to retrieve data set to be displayed

```

public ResultSet getQueryResult() {
    try {
        String query = "SELECT customercode,fullname,location,phone FROM customers";
        resultSet = statement.executeQuery(query);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return resultSet;
}

```

// Method to retrieve search data

```

public ResultSet getCustomerSearch(String text) {
    try {
        String query = "SELECT customercode,fullname,location,phone FROM customers " +
            "WHERE customercode LIKE '%" + text + "%' OR fullname LIKE '%" + text + "%' OR " +
            "location LIKE '%" + text + "%' OR phone LIKE '%" + text + "%'";
        resultSet = statement.executeQuery(query);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return resultSet;
}

```

public ResultSet getCustName(String custCode) {

```

    try {
        String query = "SELECT * FROM customers WHERE customercode='" + custCode + "'";
        resultSet = statement.executeQuery(query);
    } catch (SQLException e) {

```

```

        e.printStackTrace();
    }
    return resultSet;
}

public ResultSet getProdName(String prodCode) {
    try {
        String query = "SELECT productname,currentstock.quantity FROM products " +
            "INNER JOIN currentstock ON products.productcode=currentstock.productcode " +
            "WHERE currentstock.productcode='" + prodCode + "'";
        resultSet = statement.executeQuery(query);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return resultSet;
}

// Method to display data set in tabular form
public DefaultTableModel buildTableModel(ResultSet resultSet) throws SQLException {
    ResultSetMetaData metaData = resultSet.getMetaData();
    Vector<String> columnNames = new Vector<String>();
    int colCount = metaData.getColumnCount();

    for (int col=1; col <= colCount; col++){
        columnNames.add(metaData.getColumnName(col).toUpperCase(Locale.ROOT));
    }

    Vector<Vector<Object>> data = new Vector<Vector<Object>>();
    while (resultSet.next()) {
        Vector<Object> vector = new Vector<Object>();
        for (int col=1; col<=colCount; col++) {
            vector.add(resultSet.getObject(col));
        }
        data.add(vector);
    }
    return new DefaultTableModel(data, columnNames);
}
}

```

➤ ProductDAO

```

import javax.swing.*.*;
import java.sql.*;
import java.util.Vector;
import javax.swing.table.DefaultTableModel;
import java.util.Locale;

```



```
class ProductDTO {

    int prodID, quantity, userID;
    double costPrice, sellPrice;
    Double totalCost, totalRevenue;
    String prodCode, prodName, date, suppCode, custCode, custName, brand;

    public int getProdID() {
        return prodID;
    }

    public void setProdID(int prodID) {
        this.prodID = prodID;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public int getUserID() {
        return userID;
    }

    public void setUserID(int userID) {
        this.userID = userID;
    }

    public double getCostPrice() {
        return costPrice;
    }

    public void setCostPrice(double costPrice) {
        this.costPrice = costPrice;
    }

    public double getSellPrice() {
        return sellPrice;
    }

    public void setSellPrice(double sellPrice) {
        this.sellPrice = sellPrice;
    }

    public Double getTotalCost() {
```

```
        return totalCost;
    }

    public void setTotalCost(Double totalCost) {
        this.totalCost = totalCost;
    }

    public Double getTotalRevenue() {
        return totalRevenue;
    }

    public void setTotalRevenue(Double totalRevenue) {
        this.totalRevenue = totalRevenue;
    }

    public String getProdCode() {
        return prodCode;
    }

    public void setProdCode(String prodCode) {
        this.prodCode = prodCode;
    }

    public String getProdName() {
        return prodName;
    }

    public void setProdName(String prodName) {
        this.prodName = prodName;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getSuppCode() {
        return suppCode;
    }

    public void setSuppCode(String suppCode) {
        this.suppCode = suppCode;
    }

    public String getCustCode() {
        return custCode;
    }
```

```

    }

    public void setCustCode(String custCode) {
        this.custCode = custCode;
    }

    public String getCustName() {
        return custName;
    }

    public void setCustName(String custName) {
        this.custName = custName;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = "Dummy Brand";
    }
}

public class ProductDAO {

    Connection conn = null;
    PreparedStatement prepStatement = null;
    PreparedStatement prepStatement2 = null;
    Statement statement = null;
    Statement statement2 = null;
    ResultSet resultSet = null;

    public ProductDAO() {
        try {
            conn = new ConnectionFactory().getConn();
            statement = conn.createStatement();
            statement2 = conn.createStatement();
        } catch (Exception ex) {
        }
    }

    public ResultSet getSuppInfo() {
        try {
            String query = "SELECT * FROM suppliers";
            resultSet = statement.executeQuery(query);
        } catch (Exception e) {
        }
        return resultSet;
    }
}

```

```

public ResultSet getCustInfo() {
    try {
        String query = "SELECT * FROM customers";
        resultSet = statement.executeQuery(query);
    } catch (Exception e) {
    }
    return resultSet;
}

public ResultSet getProdStock() {
    try {
        String query = "SELECT * FROM currentstock";
        resultSet = statement.executeQuery(query);
    } catch (Exception e) {
    }
    return resultSet;
}

public ResultSet getProdInfo() {
    try {
        String query = "SELECT * FROM products";
        resultSet = statement.executeQuery(query);
    } catch (Exception e) {
    }
    return resultSet;
}

public Double getProdCost(String prodCode) {
    Double costPrice = null;
    try {
        String query = "SELECT costprice FROM products WHERE productcode='" + prodCode + "'";
        resultSet = statement.executeQuery(query);
        if (resultSet.next())
            costPrice = resultSet.getDouble("costprice");
    } catch (Exception e) {
        e.printStackTrace();
    }
    return costPrice;
}

public Double getProdSell(String prodCode) {
    Double sellPrice = null;
    try {
        String query = "SELECT sellprice FROM products WHERE productcode='" + prodCode + "'";
        resultSet = statement.executeQuery(query);
        if (resultSet.next())
            sellPrice = resultSet.getDouble("sellprice");
    } catch (Exception e) {
    }
}

```

```

        e.printStackTrace();
    }
    return sellPrice;
}

```

```

String suppCode;
public String getSuppCode(String suppName) {
    try {
        String query = "SELECT suppliercode FROM suppliers WHERE fullname='" + suppName + "'";
        resultSet = statement.executeQuery(query);
        while (resultSet.next()) {
            suppCode = resultSet.getString("suppliercode");
        }
    } catch (SQLException e) {
    }
    return suppCode;
}

```

```

String prodCode;
public String getProdCode(String prodName) {
    try {
        String query = "SELECT productcode FROM products WHERE productname='" + prodName + "'";
        resultSet = statement.executeQuery(query);
        while (resultSet.next()) {
            suppCode = resultSet.getString("productcode");
        }
    } catch (SQLException e) {
    }
    return prodCode;
}

```

```

String custCode;
public String getCustCode(String custName) {
    try {
        String query = "SELECT customercode FROM suppliers WHERE fullname='" + custName + "'";
        resultSet = statement.executeQuery(query);
        while (resultSet.next()) {
            suppCode = resultSet.getString("customercode");
        }
    } catch (SQLException e) {
    }
    return custCode;
}

```

```

// Method to check for availability of stock in Inventory
boolean flag = false;
public boolean checkStock(String prodCode) {
    try {
        String query = "SELECT * FROM currentstock WHERE productcode='" + prodCode + "'";
    }
}

```

```

        resultSet = statement.executeQuery(query);
        while (resultSet.next()) {
            flag = true;
        }
    } catch (SQLException e) {
    }
    return flag;
}

// Methods to add a new product
public void addProductDAO(ProductDTO productDTO) {
    try {
        String query = "SELECT * FROM products WHERE productname="
            + productDTO.getProdName()
            + " AND costprice="
            + productDTO.getCostPrice()
            + " AND sellprice="
            + productDTO.getSellPrice()
            + " AND brand="
            + productDTO.getBrand()
            + "";
        resultSet = statement.executeQuery(query);
        if (resultSet.next())
            JOptionPane.showMessageDialog(null, "Product has already been added.");
        else
            addFunction(productDTO);
    } catch (SQLException e) {
    }
}

public void addFunction(ProductDTO productDTO) {
    try {
        String query = "INSERT INTO products VALUES(null,?,?,?,?,?)";
        preparedStatement = (PreparedStatement) conn.prepareStatement(query);
        preparedStatement.setString(1, productDTO.getProdCode());
        preparedStatement.setString(2, productDTO.getProdName());
        preparedStatement.setDouble(3, productDTO.getCostPrice());
        preparedStatement.setDouble(4, productDTO.getSellPrice());
        preparedStatement.setString(5, productDTO.getBrand());

        String query2 = "INSERT INTO currentstock VALUES(?,?)";
        preparedStatement2 = conn.prepareStatement(query2);
        preparedStatement2.setString(1, productDTO.getProdCode());
        preparedStatement2.setInt(2, productDTO.getQuantity());

        preparedStatement.executeUpdate();
        preparedStatement2.executeUpdate();
        JOptionPane.showMessageDialog(null, "Product added and ready for sale.");
    } catch (SQLException throwables) {
    }
}

```

```
}
```

```
// Method to add a new purchase transaction
```

```
public void addPurchaseDAO(ProductDTO productDTO) {
```

```
    try {
```

```
        String query = "INSERT INTO purchaseinfo VALUES(null,?,?,?,?,?)";
```

```
        preparedStatement = conn.prepareStatement(query);
```

```
        preparedStatement.setString(1, productDTO.getSuppCode());
```

```
        preparedStatement.setString(2, productDTO.getProdCode());
```

```
        preparedStatement.setString(3, productDTO.getDate());
```

```
        preparedStatement.setInt(4, productDTO.getQuantity());
```

```
        preparedStatement.setDouble(5, productDTO.getTotalCost());
```

```
        preparedStatement.executeUpdate();
```

```
        JOptionPane.showMessageDialog(null, "Purchase log added.");
```

```
    } catch (SQLException throwables) {
```

```
    }
```

```
String prodCode = productDTO.getProdCode();
```

```
if(checkStock(prodCode)) {
```

```
    try {
```

```
        String query = "UPDATE currentstock SET quantity=quantity+? WHERE productcode=?";
```

```
        preparedStatement = conn.prepareStatement(query);
```

```
        preparedStatement.setInt(1, productDTO.getQuantity());
```

```
        preparedStatement.setString(2, prodCode);
```

```
        preparedStatement.executeUpdate();
```

```
    } catch (SQLException throwables) {
```

```
    }
```

```
}
```

```
else if (!checkStock(prodCode)) {
```

```
    try {
```

```
        String query = "INSERT INTO currentstock VALUES(?,?)";
```

```
        preparedStatement = (PreparedStatement) conn.prepareStatement(query);
```

```
        preparedStatement.setString(1, productDTO.getProdCode());
```

```
        preparedStatement.setInt(2, productDTO.getQuantity());
```

```
        preparedStatement.executeUpdate();
```

```
    } catch (SQLException throwables) {
```

```
    }
```

```
}
```

```
deleteStock();
```

```
}
```

```
// Method to update existing product details
```

```
public void editProdDAO(ProductDTO productDTO) {
```

```
    try {
```

```
        String query = "UPDATE products SET productname=?,costprice=?,sellprice=?,brand=? WHERE  
productcode=?";
```

```

        preparedStatement = (PreparedStatement) conn.prepareStatement(query);
        preparedStatement.setString(1, productDTO.getProdName());
        preparedStatement.setDouble(2, productDTO.getCostPrice());
        preparedStatement.setDouble(3, productDTO.getSellPrice());
        preparedStatement.setString(4, productDTO.getBrand());
        preparedStatement.setString(5, productDTO.getProdCode());

        String query2 = "UPDATE currentstock SET quantity=? WHERE productcode=?";
        preparedStatement2 = conn.prepareStatement(query2);
        preparedStatement2.setInt(1, productDTO.getQuantity());
        preparedStatement2.setString(2, productDTO.getProdCode());

        preparedStatement.executeUpdate();
        preparedStatement2.executeUpdate();
        JOptionPane.showMessageDialog(null, "Product details updated.");
    } catch (SQLException throwables) {
    }
}

// Methods to handle updating of stocks in Inventory upon any transaction made
public void editPurchaseStock(String code, int quantity) {
    try {
        String query = "SELECT * FROM currentstock WHERE productcode='" + code + "'";
        resultSet = statement.executeQuery(query);
        if(resultSet.next()) {
            String query2 = "UPDATE currentstock SET quantity=quantity-? WHERE productcode=?";
            preparedStatement = conn.prepareStatement(query2);
            preparedStatement.setInt(1, quantity);
            preparedStatement.setString(2, code);
            preparedStatement.executeUpdate();
        }
    } catch (SQLException throwables) {
    }
}

public void editSoldStock(String code, int quantity) {
    try {
        String query = "SELECT * FROM currentstock WHERE productcode='" + code + "'";
        resultSet = statement.executeQuery(query);
        if(resultSet.next()) {
            String query2 = "UPDATE currentstock SET quantity=quantity+? WHERE productcode=?";
            preparedStatement = conn.prepareStatement(query2);
            preparedStatement.setInt(1, quantity);
            preparedStatement.setString(2, code);
            preparedStatement.executeUpdate();
        }
    } catch (SQLException throwables) {
    }
}

public void deleteStock() {

```



```

    try {
        String query = "DELETE FROM currentstock WHERE productcode NOT IN(SELECT productcode FROM
purchaseinfo)";
        String query2 = "DELETE FROM salesinfo WHERE productcode NOT IN(SELECT productcode FROM
products)";
        statement.executeUpdate(query);
        statement.executeUpdate(query2);
    } catch (SQLException throwables) {
    }
}

// Method to permanently delete a product from inventory
public void deleteProductDAO(String code) {
    try {
        String query = "DELETE FROM products WHERE productcode=?";
        prepStatement = conn.prepareStatement(query);
        prepStatement.setString(1, code);

        String query2 = "DELETE FROM currentstock WHERE productcode=?";
        prepStatement2 = conn.prepareStatement(query2);
        prepStatement2.setString(1, code);

        prepStatement.executeUpdate();
        prepStatement2.executeUpdate();

        JOptionPane.showMessageDialog(null, "Product has been removed.");
    } catch (SQLException e){
    }
    deleteStock();
}

public void deletePurchaseDAO(int ID){
    try {
        String query = "DELETE FROM purchaseinfo WHERE purchaseID=?";
        prepStatement = conn.prepareStatement(query);
        prepStatement.setInt(1, ID);
        prepStatement.executeUpdate();

        JOptionPane.showMessageDialog(null, "Transaction has been removed.");
    } catch (SQLException e){
    }
    deleteStock();
}

// Products data set retrieval for display
public ResultSet getQueryResult() {
    try {
        String query = "SELECT productcode,productname,costprice,sellprice,brand FROM products ORDER BY
pid";

```

```

        resultSet = statement.executeQuery(query);
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    return resultSet;
}

```

// Purchase table data set retrieval

```

public ResultSet getPurchaseInfo() {
    try {
        String query = "SELECT PurchaseID,purchaseinfo.ProductCode,ProductName,Quantity,Totalcost " +
            "FROM purchaseinfo INNER JOIN products " +
            "ON products.productcode=purchaseinfo.productcode ORDER BY purchaseid;";
        resultSet = statement.executeQuery(query);
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    return resultSet;
}

```

// Stock table data set retrieval

```

public ResultSet getCurrentStockInfo() {
    try {
        String query = ""
            SELECT currentstock.ProductCode,products.ProductName,
            currentstock.Quantity,products.CostPrice,products.SellPrice
            FROM currentstock INNER JOIN products
            ON currentstock.productcode=products.productcode;
        """;
        resultSet = statement.executeQuery(query);
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    return resultSet;
}

```

// Search method for products

```

public ResultSet getProductSearch(String text) {
    try {
        String query = "SELECT productcode,productname,costprice,sellprice,brand FROM products " +
            "WHERE productcode LIKE '%" + text + "%' OR productname LIKE '%" + text + "%' OR brand LIKE '%" + text + "%'";
        resultSet = statement.executeQuery(query);
    } catch (SQLException e) {
    }
    return resultSet;
}

```

```

public ResultSet getProdFromCode(String text) {

```

```

try {
    String query = "SELECT productcode,productname,costprice,sellprice,brand FROM products " +
        "WHERE productcode='" +text+" ' LIMIT 1";
    resultSet = statement.executeQuery(query);
} catch (SQLException e) {
    e.printStackTrace();
}
return resultSet;
}

// Search method for purchase logs
public ResultSet getPurchaseSearch(String text) {
    try {
        String query = "SELECT PurchaseID,purchaseinfo.productcode,products.productname,quantity,totalcost "
+
            "FROM purchaseinfo INNER JOIN products ON purchaseinfo.productcode=products.productcode " +
            "INNER JOIN suppliers ON purchaseinfo.suppliercode=suppliers.suppliercode" +
            "WHERE PurchaseID LIKE '%" +text+"%' OR productcode LIKE '%" +text+"%' OR productname LIKE
 '%" +text+"%' " +
            "OR suppliers.fullname LIKE '%" +text+"%' OR purchaseinfo.suppliercode LIKE '%" +text+"%' " +
            "OR date LIKE '%" +text+"%' ORDER BY purchaseid";
        resultSet = statement.executeQuery(query);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return resultSet;
}

public ResultSet getProdName(String code) {
    try {
        String query = "SELECT productname FROM products WHERE productcode='" +code+ "'";
        resultSet = statement.executeQuery(query);
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    return resultSet;
}

public String getSuppName(int ID) {
    String name = null;
    try {
        String query = "SELECT fullname FROM suppliers " +
            "INNER JOIN purchaseinfo ON suppliers.suppliercode=purchaseinfo.suppliercode " +
            "WHERE purchaseid='" +ID+ "'";
        resultSet = statement.executeQuery(query);
        if (resultSet.next())
            name = resultSet.getString("fullname");
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
}

```

```

    }
    return name;
}

public String getCustName(int ID) {
    String name = null;
    try {
        String query = "SELECT fullname FROM customers " +
            "INNER JOIN salesinfo ON customers.customercode=salesinfo.customercode " +
            "WHERE salesid='" + ID + "'";
        resultSet = statement.executeQuery(query);
        if (resultSet.next())
            name = resultSet.getString("fullname");
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    return name;
}

public String getPurchaseDate(int ID) {
    String date = null;
    try {
        String query = "SELECT date FROM purchaseinfo WHERE purchaseid='" + ID + "'";
        resultSet = statement.executeQuery(query);
        if (resultSet.next())
            date = resultSet.getString("date");
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    return date;
}

public String getSaleDate(int ID) {
    String date = null;
    try {
        String query = "SELECT date FROM salesinfo WHERE salesid='" + ID + "'";
        resultSet = statement.executeQuery(query);
        if (resultSet.next())
            date = resultSet.getString("date");
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    return date;
}

```

```

// Method to display product-related data set in tabular form
public DefaultTableModel buildTableModel(ResultSet resultSet) throws SQLException {
    ResultSetMetaData metaData = resultSet.getMetaData();
    Vector<String> columnNames = new Vector<String>();

```

```

int colCount = metaData.getColumnCount();

for (int col=1; col <= colCount; col++){
    columnNames.add(metaData.getColumnName(col).toUpperCase(Locale.ROOT));
}

Vector<Vector<Object>> data = new Vector<Vector<Object>>();
while (resultSet.next()) {
    Vector<Object> vector = new Vector<Object>();
    for (int col=1; col<=colCount; col++) {
        vector.add(resultSet.getObject(col));
    }
    data.add(vector);
}
return new DefaultTableModel(data, columnNames);
}
}

```

➤ Login page

```

import java.awt.*;
import javax.swing.*;

public class LoginPage extends javax.swing.JFrame {
    public LoginPage() {
        initComponents();
    }

    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        userText = new javax.swing.JTextField();
        passText = new javax.swing.JPasswordField();
        jLabel3 = new javax.swing.JLabel();
        loginButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setBounds(new java.awt.Rectangle(400, 100, 0, 0));
        setName("loginFrame");

        jLabel1.setFont(new java.awt.Font("MV Boli", 0, 14));
        jLabel1.setText("Username:");

        jLabel2.setFont(new java.awt.Font("MV Boli", 0, 14));
        jLabel2.setText("Password:");

        jLabel3.setFont(new java.awt.Font("MV Boli", Font.BOLD, 20));
        jLabel3.setForeground(Color.BLACK);
    }
}

```

```

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel3.setText("MANAGE INVENTORY");

loginButton.setText("LOGIN");
loginButton.setBackground(new Color(0X05386B));
loginButton.setForeground(new Color(0XEDF5E1));
loginButton.setFocusable(false);
loginButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
loginButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        loginButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
getContentPane().setBackground(new Color(0X8EE4AF));
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(47, 47, 47)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 74,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(userText, javax.swing.GroupLayout.DEFAULT_SIZE, 204, Short.MAX_VALUE))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 74,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(passText)))
            .addGap(72, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addGap(121, 121, 121))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 284,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(47, 47, 47)))
    );
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(53, 53, 53)

```

```

        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(48, 48, 48)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(userText, javax.swing.GroupLayout.DEFAULT_SIZE, 31, Short.MAX_VALUE)
            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(passText, javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(1, 1, 1)))
        .addGap(27, 27, 27)
        .addComponent(loginButton, javax.swing.GroupLayout.PREFERRED_SIZE, 37,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(80, Short.MAX_VALUE))
    );

    pack();
}

```

```

private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String username = userText.getText();
    String password = passText.getText();

    if (new ConnectionFactory().checkLogin(username, password, "ADMINISTRATOR")){
        dispose();
        new Dashboard(username, "ADMINISTRATOR");
    } else {
        JOptionPane.showMessageDialog(
            null,
            "Invalid username or password.");
    }
}

```

```

public static void main(String[] args) {
    new LoginPage().setVisible(true);
}

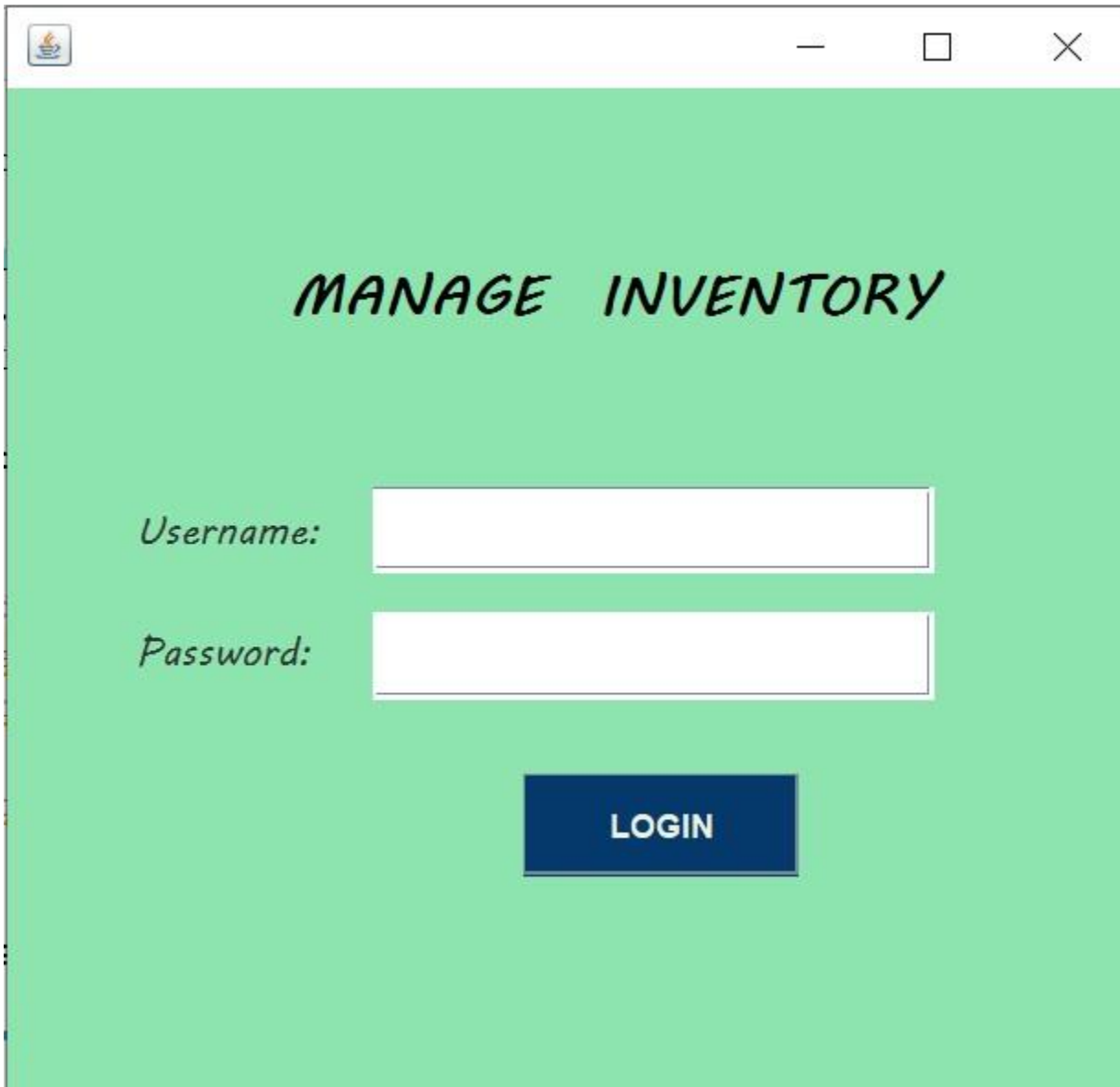
```

```

javax.swing.JLabel jLabel1;
javax.swing.JLabel jLabel2;
javax.swing.JLabel jLabel3;
javax.swing.JButton loginButton;
javax.swing.JPasswordField passText;

```

```
javax.swing.JTextField userText;  
}
```



The image shows a Java Swing window titled "MANAGE INVENTORY". The window has a light green background and a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The title bar also features a small icon on the left. The main content area of the window contains the following elements:

- The title "MANAGE INVENTORY" is centered at the top in a large, bold, black, monospace-style font.
- Below the title, there are two text input fields. The first field is preceded by the label "Username:" and the second field is preceded by the label "Password:". Both labels are in a black, monospace-style font.
- Below the input fields, there is a dark blue rectangular button with the word "LOGIN" in white, bold, uppercase letters.

➤ Dashboard.java

```
import java.awt.CardLayout;
import java.awt.Color;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class Dashboard extends javax.swing.JFrame {
    CardLayout layout;

    public Dashboard(String username, String userType) {
        initComponents();
        navPanel.setVisible(true);
        menuPanel.setVisible(true);
        layout = new CardLayout();

        displayPanel.setLayout(layout);
        displayPanel.add("Home", new HomePage());
        displayPanel.add("Customers", new CustomerPage());
        displayPanel.add("Products", new ProductPage(username, this));
        displayPanel.add("Current Stock", new CurrentStockPage(username));
        displayPanel.add("Sales", new PurchasePage(this));

        this.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                super.windowClosing(e);
            }
        });

        setTitle("Inventory Manager");
        setVisible(true);
    }

    public void addHomePage(){
        layout.show(displayPanel, "Home");
    }
    public void addCustPage() {
        layout.show(displayPanel, "Customers");
    }
    public void addProdPage() {
        layout.show(displayPanel, "Products");
    }
    public void addStockPage() {
        layout.show(displayPanel, "Current Stock");
    }
    public void addPurchasePage() {
        layout.show(displayPanel, "Sales");
    }
}
```

```

private void initComponents() {

    mainPanel = new javax.swing.JPanel();
    menuPanel = new javax.swing.JPanel();
    navPanel = new javax.swing.JPanel();
    prodButton = new javax.swing.JButton();
    stockButton = new javax.swing.JButton();
    custButton = new javax.swing.JButton();
    purchaseButton = new javax.swing.JButton();
    displayPanel = new javax.swing.JPanel();
    userPanel = new javax.swing.JPanel();
    jMenuBar1 = new javax.swing.JMenuBar();

    mainPanel.setBackground(new Color(0X5CDB95));
    menuPanel.setBackground(new Color(0X5CDB95));
    navPanel.setBackground(new Color(0X8EE4AF));
    userPanel.setBackground(new Color(0X5CDB95));

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setBounds(new java.awt.Rectangle(200, 100, 0, 0));

    menuPanel.setPreferredSize(new java.awt.Dimension(120, 26));

    javax.swing.GroupLayout menuPanelLayout = new javax.swing.GroupLayout(menuPanel);
    menuPanel.setLayout(menuPanelLayout);
    menuPanelLayout.setHorizontalGroup(
        menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 125, Short.MAX_VALUE)
    );
    menuPanelLayout.setVerticalGroup(
        menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 50, Short.MAX_VALUE)
    );

    navPanel.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    prodButton.setText("Products");
    prodButton.setBackground(new Color(0X05386B));
    prodButton.setForeground(new Color(0XEDF5E1));
    prodButton.setFocusable(false);
    prodButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    prodButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            prodButtonActionPerformed(evt);
        }
    });

    stockButton.setText("Current Stock");

```

```

stockButton.setBackground(new Color(0X05386B));
stockButton.setForeground(new Color(0XEDF5E1));
stockButton.setFocusable(false);
stockButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
stockButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        stockButtonActionPerformed(evt);
    }
});

```

```

custButton.setText("Customers");
custButton.setBackground(new Color(0X05386B));
custButton.setForeground(new Color(0XEDF5E1));
custButton.setFocusable(false);
custButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
custButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        custButtonActionPerformed(evt);
    }
});

```

```

purchaseButton.setText("Sales");
purchaseButton.setBackground(new Color(0X05386B));
purchaseButton.setForeground(new Color(0XEDF5E1));
purchaseButton.setFocusable(false);
purchaseButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
purchaseButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        purchaseButtonActionPerformed(evt);
    }
});

```

```

javax.swing.GroupLayout navPanelLayout = new javax.swing.GroupLayout(navPanel);
navPanel.setLayout(navPanelLayout);
navPanelLayout.setHorizontalGroup(
    navPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, navPanelLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(purchaseButton, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGap(10, 10, 0)
            .addComponent(prodButton, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGap(10, 10, 0)
            .addComponent(stockButton, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGap(10, 10, 0)
            .addComponent(custButton, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGap(10, 10, 0)
        )
);

```

```

navPanelLayout.setVerticalGroup(
    navPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(navPanelLayout.createSequentialGroup()
            .addGap(44, 44, 44)
            .addComponent(prodButton, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(43, 43, 43)
            .addComponent(stockButton, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(45, 45, 45)
            .addComponent(custButton, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(44, 44, 44)
            .addComponent(purchaseButton, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(45, Short.MAX_VALUE))
        );

displayPanel.setLayout(new java.awt.CardLayout());

javax.swing.GroupLayout userPanelLayout = new javax.swing.GroupLayout(userPanel);
userPanel.setLayout(userPanelLayout);
userPanelLayout.setHorizontalGroup(
    userPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 780, Short.MAX_VALUE)
    );
userPanelLayout.setVerticalGroup(
    userPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 38, Short.MAX_VALUE)
    );

javax.swing.GroupLayout mainPanelLayout = new javax.swing.GroupLayout(mainPanel);
mainPanel.setLayout(mainPanelLayout);
mainPanelLayout.setHorizontalGroup(
    mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(mainPanelLayout.createSequentialGroup()
            .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(navPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(menuPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 125, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(displayPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(userPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
        );
mainPanelLayout.setVerticalGroup(
    mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

        .addGroup(mainPanelLayout.createSequentialGroup())
        .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(userPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(menuPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(displayPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 495,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(navPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    setJMenuBar(jMenuBar1);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );

    pack();
}

private void custButtonActionPerformed(java.awt.event.ActionEvent evt) {
    addCustPage();
}

private void stockButtonActionPerformed(java.awt.event.ActionEvent evt) {
    addStockPage();
}

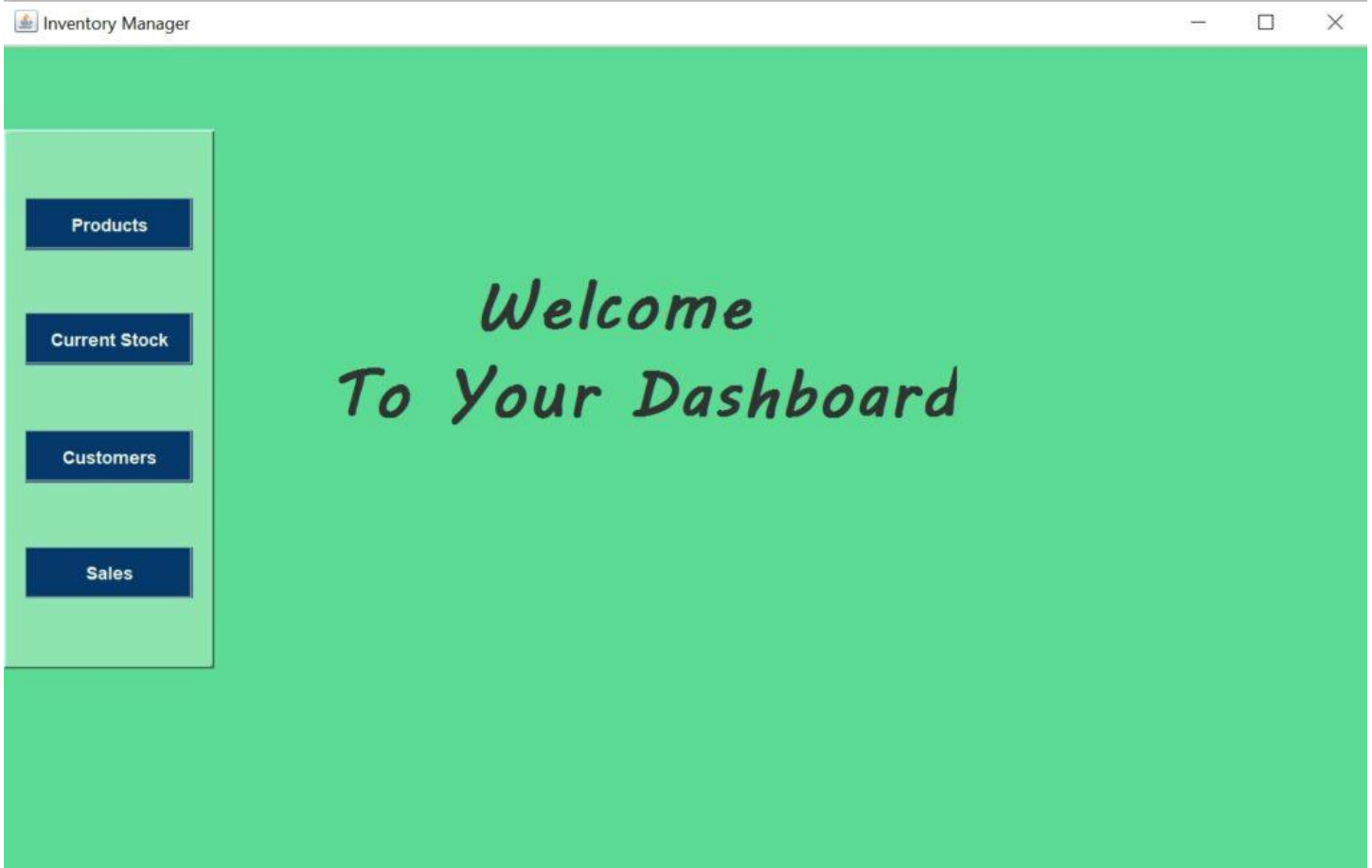
private void prodButtonActionPerformed(java.awt.event.ActionEvent evt) {
    addProdPage();
}

private void purchaseButtonActionPerformed(java.awt.event.ActionEvent evt) {
    addPurchasePage();
}

```

```
javax.swing.JButton custButton;  
javax.swing.JPanel displayPanel;  
javax.swing.JMenuBar jMenuBar1;  
javax.swing.JPanel mainPanel;  
javax.swing.JPanel menuPanel;  
javax.swing.JPanel navPanel;  
javax.swing.JButton prodButton;  
javax.swing.JButton purchaseButton;  
javax.swing.JButton stockButton;  
javax.swing.JPanel userPanel;
```

```
}
```



> Home page

```
import java.awt.*;

public class HomePage extends javax.swing.JPanel {

    public HomePage() {
        initComponents();
    }

    private void initComponents() {

        setBackground(new Color(0X5CDB95));

        welcomeLabel = new javax.swing.JLabel();
        welcomeLabel1 = new javax.swing.JLabel();

        welcomeLabel.setFont(new java.awt.Font("MV Boli", Font.BOLD, 45));
        welcomeLabel.setText("Welcome");

        welcomeLabel1.setFont(new java.awt.Font("MV Boli", Font.BOLD, 45));
        welcomeLabel1.setText("To Your Dashboard");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(170, 170, 170)
                    .addComponent(welcomeLabel)
                    .addGap(71, 71, 71)
                    .addComponent(welcomeLabel1))
                .addGroup(layout.createSequentialGroup()
                    .addGap(83, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(96, 96, 96)
                    .addComponent(welcomeLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(welcomeLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(169, Short.MAX_VALUE))
        );
    }
    javax.swing.JLabel welcomeLabel;
```

```
javax.swing.JLabel welcomeLabel1;  
}
```



*Welcome
To Your Dashboard*

> product page

```
import java.awt.Color;
```

```
import javax.swing.*;
```

```
import java.sql.SQLException;
```

```
public class ProductPage extends javax.swing.JPanel {
```

```
    ProductDTO productDTO;
```

```
    String username = null;
```

```
    String supplier = null;
```

```
    int userID;
```

```
    Dashboard dashboard;
```

```
    public ProductPage() {
```

```
    }
```

```
    public ProductPage(String username, Dashboard dashboard){
```

```
        initComponents();
```

```
        this.username = username;
```

```
        this.dashboard = dashboard;
```

```
        loadDataSet();
```

```
    }
```

```
    private void initComponents() {
```

```
        jLabel1 = new javax.swing.JLabel();
```

```
        jSeparator1 = new javax.swing.JSeparator();
```

```
        entryPanel = new javax.swing.JPanel();
```

```
        jLabel2 = new javax.swing.JLabel();
```

```
        jLabel3 = new javax.swing.JLabel();
```

```
        jLabel5 = new javax.swing.JLabel();
```

```
jLabel6 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
jLabel8 = new javax.swing.JLabel();
codeText = new javax.swing.JTextField();
nameText = new javax.swing.JTextField();
quantityText = new javax.swing.JTextField();
costText = new javax.swing.JTextField();
sellText = new javax.swing.JTextField();
brandText = new javax.swing.JTextField();
addButton = new javax.swing.JButton();
jScrollPane1 = new javax.swing.JScrollPane();
productTable = new javax.swing.JTable();

setBackground(new Color(0X8EE4AF));
jLabel1.setFont(new java.awt.Font("MV Boli", 0, 15));
jLabel1.setText("PRODUCTS");

entryPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter Product Details"));

jLabel2.setText("Product Code:");

jLabel3.setText("Product Name:");

jLabel5.setText("Quantity:");

jLabel6.setText("Cost Price:");

jLabel7.setText("Selling Price:");

jLabel8.setText("Brand:");

addButton.setText("Add");
```

```

addButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));

addButton.setBackground(new Color(0X05386B));

addButton.setForeground(new Color(0XEDF5E1));

addButton.setFocusable(false);

addButton.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        addButtonActionPerformed(evt);

    }

});

```

```

javax.swing.GroupLayout entryPanelLayout = new javax.swing.GroupLayout(entryPanel);
entryPanel.setLayout(entryPanelLayout);
entryPanel.setBackground(new Color(0X8EE4AF));
entryPanelLayout.setHorizontalGroup(

    entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(entryPanelLayout.createSequentialGroup()

            .addGap(10, 10, 10)

            .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(codeText))

            .addGap(10, 10, 10)

            .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(nameText))

            .addGap(10, 10, 10)

            .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(quantityText))

```

```

        .addGroup(entryPanelLayout.createSequentialGroup())

        .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(costText))

    .addGroup(entryPanelLayout.createSequentialGroup())

    .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addComponent(sellText))

    .addGroup(entryPanelLayout.createSequentialGroup())

    .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addComponent(brandText)

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, entryPanelLayout.createSequentialGroup())

            .addGap(0, 15, Short.MAX_VALUE)

            .addComponent(addButton, javax.swing.GroupLayout.PREFERRED_SIZE, 104,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(28, 28, 28))))))

    .addContainerGap())

);

entryPanelLayout.setVerticalGroup(

    entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(entryPanelLayout.createSequentialGroup())

            .addContainerGap()

            .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(codeText, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

```

```

        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(nameText, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(quantityText, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(costText, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(sellText, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(brandText, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(addButton, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addContainerGap(140, Short.MAX_VALUE))

    );

```

```

productTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

```

```
{null, null, null, null},  
{null, null, null, null},  
{null, null, null, null},  
{null, null, null, null}  
  
},  
  
new String [] {  
    "Title 1", "Title 2", "Title 3", "Title 4"  
}  
  
));  
  
productTable.setBackground(new Color(0XEDF5E1));  
  
productTable.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));  
  
productTable.addMouseListener(new java.awt.event.MouseAdapter() {  
    public void mouseClicked(java.awt.event.MouseEvent evt) {  
        productTableMouseClicked(evt);  
    }  
});  
  
jScrollPane1.setViewportViewView(productTable);  
  
  
  
  
jSeparator1.setBackground(new Color(0X05386B));  
  
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);  
this.setLayout(layout);  
  
layout.setHorizontalGroup(  
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
        .addGroup(layout.createSequentialGroup()  
            .addGap(19, 19, 19)  
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                .addComponent(jLabel1)  
                .addComponent(jSeparator1))  
            .addContainerGap(77, Short.MAX_VALUE))  
        .addGroup(layout.createSequentialGroup()  
            .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 126,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
            .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 596, Short.MAX_VALUE)  
            .addComponent(jSeparator2)  
            .addContainerGap())  
        .addGroup(layout.createSequentialGroup()  
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  
                .addGroup(layout.createSequentialGroup()  
                    .addComponent(jTextField1)  
                    .addGap(19, 19, 19)
```

```

        .addComponent(jScrollPane1)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(entryPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))

        .addContainerGap()

    );

    layout.setVerticalGroup(

        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(layout.createSequentialGroup()

            .addContainerGap()

            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

                .addComponent(entryPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))

            .addContainerGap()

        );
}

```

```

private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    productDTO = new ProductDTO();

    if (nameText.getText().equals("") || costText.getText().equals("")
        || sellText.getText().equals("") || brandText.getText().equals(""))
        JOptionPane.showMessageDialog(null, "Please enter all the required details.");
    else {
        productDTO.setProdCode(codeText.getText());

        productDTO.setProdName(nameText.getText());

        productDTO.setDate("02/01/2020");
    }
}

```

```

        productDTO.setQuantity(Integer.parseInt(quantityText.getText()));
        productDTO.setCostPrice(Double.parseDouble(costText.getText()));
        productDTO.setSellPrice(Double.parseDouble(sellText.getText()));
        productDTO.setBrand(brandText.getText());
        productDTO.setUserID(userID);

        new ProductDAO().addProductDAO(productDTO);
        loadDataSet();
    }
}

//static String productName;

private void productTableMouseClicked(java.awt.event.MouseEvent evt) {
    int row = productTable.getSelectedRow();
    int col = productTable.getColumnCount();

    Object[] data = new Object[col];
    for (int i=0; i<col; i++)
        data[i] = productTable.getValueAt(row, i);

    codeText.setText(data[0].toString());
    nameText.setText(data[1].toString());
    costText.setText(data[2].toString());
    sellText.setText(data[3].toString());
    brandText.setText(data[4].toString());
}

public void loadDataSet() {
    try {
        ProductDAO productDAO = new ProductDAO();
        productTable.setModel(productDAO.buildTableModel(productDAO.getQueryResult()));
    } catch (SQLException throwables) {

```



```
        throwables.printStackTrace();
    }
}
```

// Method to display search result in table

```
public void loadSearchData(String text) {
    try {
        ProductDAO productDAO = new ProductDAO();
        productTable.setModel(productDAO.buildTableModel(productDAO.getProductSearch(text)));
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
}
```

```
javax.swing.JButton addButton;
javax.swing.JTextField brandText;
javax.swing.JTextField codeText;
javax.swing.JTextField costText;
javax.swing.JPanel entryPanel;
javax.swing.JLabel jLabel1;
javax.swing.JLabel jLabel2;
javax.swing.JLabel jLabel3;
javax.swing.JLabel jLabel5;
javax.swing.JLabel jLabel6;
javax.swing.JLabel jLabel7;
javax.swing.JLabel jLabel8;
javax.swing.JScrollPane jScrollPane1;
javax.swing.JSeparator jSeparator1;
javax.swing.JTextField nameText;
javax.swing.JTable productTable;
javax.swing.JTextField quantityText;
javax.swing.JTextField sellText;
```

}

PRODUCTS

PRODUCTCO...	PRODUCTNAME	COSTPRICE	SELLPRICE	BRAND
prod1	Laptop	85000.0	90000.0	Dell
prod2	Laptop	70000.0	72000.0	HP
prod3	Mobile	60000.0	64000.0	Apple
prod4	Mobile	50000.0	51000.0	Samsung
prod5	Charger	2000.0	2100.0	Apple
prod6	Mouse	1700.0	1900.0	Dell
prod7	Power Adapter	3000.0	3500.0	Dell
prod8	Smart Watch	15000.0	17000.0	Apple

Enter Product Details

Product Code:

Product Na...

Quantity:

Cost Price:

Selling Price:

Brand:

Add

> current stock page

```
import java.awt.Color;
```

```
import java.sql.SQLException;
```

```
public class CurrentStockPage extends javax.swing.JPanel {
```

```
    public CurrentStockPage(String username) {
```

```
        initComponents();
```

```
        loadDataSet();
```

```
    }
```

```
    private void initComponents() {
```

```
        jLabel1 = new javax.swing.JLabel();
```

```
        jSeparator1 = new javax.swing.JSeparator();
```

```
        jScrollPane1 = new javax.swing.JScrollPane();
```

```
        stockTable = new javax.swing.JTable();
```

```
        jLabel1.setFont(new java.awt.Font("MV Boli", 0, 15));
```

```
        jLabel1.setText("CURRENT STOCK");
```

```
        jLabel1.setToolTipText("");
```

```
        setBackground(new Color(0X8EE4AF));
```

```
        stockTable.setModel(new javax.swing.table.DefaultTableModel(  
            new Object [][] {
```

```
                {null, null, null, null},
```

```
                {null, null, null, null},
```

```
                {null, null, null, null},
```

```
                {null, null, null, null}
```

```
            },
```

```
            new String [] {
```

```
                "Title 1", "Title 2", "Title 3", "Title 4"
```

```
            },
```

```

    }

));
jScrollPane1.setViewportViewView(stockTable);
jSeparator1.setBackground(new Color(0X05386B));
stockTable.setBackground(new Color(0XEDF5E1));

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
this.setLayout(layout);
this.setBackground(new Color(0X8EE4AF));
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(jSeparator1)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 701, Short.MAX_VALUE)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 165,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(0, 0, Short.MAX_VALUE)))
            .addGap())
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap()
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```
        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 330,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
        .addContainerGap(88, Short.MAX_VALUE)  
    );  
}
```

```
public void loadDataSet() {  
    try {  
        ProductDAO productDAO = new ProductDAO();  
        stockTable.setModel(productDAO.buildTableModel(productDAO.getCurrentStockInfo()));  
    } catch (SQLException e) {  
    }  
}
```

```
javax.swing.JLabel jLabel1;  
javax.swing.JScrollPane jScrollPane1;  
javax.swing.JSeparator jSeparator1;  
javax.swing.JTable stockTable;  
}
```

CURRENT STOCK

PRODUCTCODE	PRODUCTNAME	QUANTITY	COSTPRICE	SELLPRICE
prod1	Laptop	146	85000.0	90000.0
prod2	Laptop	100	70000.0	72000.0
prod3	Mobile	202	60000.0	64000.0
prod4	Mobile	172	50000.0	51000.0
prod5	Charger	500	2000.0	2100.0
prod6	Mouse	500	1700.0	1900.0
prod7	Power Adapter	10	3000.0	3500.0
prod8	Smart Watch	20	15000.0	17000.0

➤ Customer page

```
import java.awt.Color;
import java.sql.SQLException;
import javax.swing.JOptionPane;
```

```
public class CustomerPage extends javax.swing.JPanel {
```

```
    public CustomerPage() {
        initComponents();
        loadDataSet();
    }
```

```
    private void initComponents() {
```

```
        jLabel1 = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        entryPanel = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        phoneText = new javax.swing.JTextField();
        locationText = new javax.swing.JTextField();
        codeText = new javax.swing.JTextField();
        nameText = new javax.swing.JTextField();
        creditText = new javax.swing.JTextField();
        debitText = new javax.swing.JTextField();
        addButton = new javax.swing.JButton();
        deleteButton = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        custTable = new javax.swing.JTable();

        setBackground(new Color(0X8EE4AF));
        jSeparator1.setBackground(new Color(0X05386B));
        jLabel1.setFont(new java.awt.Font("MV Boli", 0, 15));
        jLabel1.setText("CUSTOMERS");
```

```
        entryPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter Customer Details"));
```

```
        jLabel2.setText("Customer Code:");
```

```
        jLabel3.setText("Full Name:");
```

```
        jLabel4.setText("Location:");
```

```
        jLabel5.setText("Contact:");
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
```



```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(phoneText, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(creditText)
            .addComponent(debitText)
            .addComponent(locationText)
            .addComponent(codeText)
            .addComponent(nameText)))
        .addGroup(entryPanelLayout.createSequentialGroup())
        .addGap(41, 41, 41)
        .addComponent(addButton)
        .addGap(18, 18, 18)
        .addComponent(deleteButton)
        .addGap(0, 0, Short.MAX_VALUE)))
    .addContainerGap())
);
entryPanelLayout.setVerticalGroup(
    entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(entryPanelLayout.createSequentialGroup())
        .addContainerGap()
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(codeText, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(nameText, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(locationText, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(phoneText, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(debitText, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE))

```

```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(creditText, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(addButton)
            .addComponent(deleteButton))
        .addContainerGap(54, Short.MAX_VALUE)
    );

    custTable.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3", "Title 4"
        }
    ));
    custTable.setBackground(new Color(0XEDF5E1));
    custTable.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            custTableMouseClicked(evt);
        }
    });
    jScrollPane1.setViewportViewView(custTable);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
    this.setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 122,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                            .addComponent(jSeparator1)
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 451, Short.MAX_VALUE)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

        .addComponent(entryPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap())
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(entryPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
            .addContainerGap(67, Short.MAX_VALUE))
        );
}

private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (codeText.getText().equals("") || nameText.getText().equals("")
        || locationText.getText().equals("") || phoneText.getText().equals(""))
        JOptionPane.showMessageDialog(this, "Please enter all the required details.");
    else {
        CustomerDTO customerDTO = new CustomerDTO();
        customerDTO.setCustCode(codeText.getText());
        customerDTO.setFullName(nameText.getText());
        customerDTO.setLocation(locationText.getText());
        customerDTO.setPhone(phoneText.getText());
        new CustomerDAO().addCustomerDAO(customerDTO);
        loadDataSet();
    }
}

private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new CustomerDAO().deleteCustomerDAO(custTable.getValueAt(custTable.getSelectedRow(),0).toString());
    loadDataSet();
}

private void custTableMouseClicked(java.awt.event.MouseEvent evt) {
    int row = custTable.getSelectedRow();
    int col = custTable.getColumnCount();
    Object[] data = new Object[col];

    for (int i=0; i<col; i++)
        data[i] = custTable.getValueAt(row, i);
}

```

```

        codeText.setText((String) data[0]);
        nameText.setText((String) data[1]);
        locationText.setText((String) data[2]);
        phoneText.setText((String) data[3]);
    }

    public void loadDataSet() {
        try {
            CustomerDAO customerDAO = new CustomerDAO();
            custTable.setModel(customerDAO.buildTableModel(customerDAO.getQueryResult()));
        } catch (SQLException e) {
        }
    }

    public void loadSearchData(String text) {
        try {
            CustomerDAO customerDAO = new CustomerDAO();
            custTable.setModel(customerDAO.buildTableModel(customerDAO.getCustomerSearch(text)));
        } catch (SQLException e) {
        }
    }

    javax.swing.JButton addButton;
    javax.swing.JTextField codeText;
    javax.swing.JTextField creditText;
    javax.swing.JTable custTable;
    javax.swing.JTextField debitText;
    javax.swing.JButton deleteButton;
    javax.swing.JPanel entryPanel;
    javax.swing.JLabel jLabel1;
    javax.swing.JLabel jLabel2;
    javax.swing.JLabel jLabel3;
    javax.swing.JLabel jLabel4;
    javax.swing.JLabel jLabel5;
    javax.swing.JLabel jLabel6;
    javax.swing.JLabel jLabel7;
    javax.swing.JScrollPane jScrollPane1;
    javax.swing.JSeparator jSeparator1;
    javax.swing.JTextField locationText;
    javax.swing.JTextField nameText;
    javax.swing.JTextField phoneText;
}

```

CUSTOMERS

CUSTOMERCODE	FULLNAME	LOCATION	PHONE
vip1	John Seed	New York	9818562354
vip2	Jacob Seed	Texas	9650245489
std1	Ajay Kumar	Mumbai	9236215622
std2	Astha Walia	Chandigarh	8854612478
vip3	Madhu Chitkara	Chandigarh	9826546182

Enter Customer Details

Customer Code:

Full Name:

Location:

Contact:

Debit Amount:

Credit Amount:

Add

Delete

➤ Seles page

```
import java.awt.Color;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class PurchasePage extends javax.swing.JPanel {

    ProductDTO productDTO;

    public PurchasePage(Dashboard dashboard) {
        initComponents();
        loadDataSet();
    }

    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        jPanel1 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        codeText = new javax.swing.JTextField();
        nameText = new javax.swing.JTextField();
        quantityText = new javax.swing.JTextField();
        costText = new javax.swing.JTextField();
        sellText = new javax.swing.JTextField();
        purchaseButton = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        purchaseTable = new javax.swing.JTable();

        setBackground(new Color(0X8EE4AF));
        jSeparator1.setBackground(new Color(0X05386B));

        jLabel1.setFont(new java.awt.Font("MV Boli", 0, 15));
        jLabel1.setText("SALES");

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Sales Info"));

        jLabel3.setText("Product Code:");

        jLabel4.setText("Product Name:");

        jLabel6.setText("Quantity:");

        jLabel7.setText("Cost Price:");
```

```

jLabel8.setText("Selling Price:");

codeText.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        codeTextKeyReleased(evt);
    }
});

purchaseButton.setText("Sales");
purchaseButton.setBackground(new Color(0X05386B));
purchaseButton.setForeground(new Color(0XEDF5E1));
purchaseButton.setFocusable(false);
purchaseButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        purchaseButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1. setBackground(new Color(0X8EE4AF));
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(10, 10, 10)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 102,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(codeText, javax.swing.GroupLayout.DEFAULT_SIZE, 173, Short.MAX_VALUE))
            .addGap(10, 10, 10)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 102,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(nameText))
            .addGap(10, 10, 10)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 102,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(quantityText))
            .addGap(10, 10, 10)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 102,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(costText))
            .addGap(10, 10, 10)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 102,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(purchaseButton))
        )
);

```

```

        .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 102,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(sellText)))
        .addContainerGap())
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup())
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(purchaseButton, javax.swing.GroupLayout.PREFERRED_SIZE, 113,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(34, 34, 34))
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup())
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(codeText, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(nameText, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(quantityText, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(costText, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(sellText, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(purchaseButton, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(29, 29, 29))
    );

```



```

purchaseTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null}
    },
    new String [] {
        "Title 1", "Title 2", "Title 3", "Title 4"
    }
));

purchaseTable.setBackground(new Color(0XEDF5E1));

purchaseTable.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        purchaseTableMouseClicked(evt);
    }
});
jScrollPane1.setViewportView(purchaseTable);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
this.setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jSeparator1)
        .addGroup(layout.createSequentialGroup()
            .addGap(10, 10, 10)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 112,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(10, 10, 10)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 459, Short.MAX_VALUE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(10, 10, 10)
            )
        );
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(10, 10, 10)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 43,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
}

```

```

private void purchaseTableMouseClicked(java.awt.event.MouseEvent evt) {

```

```

    int row = purchaseTable.getSelectedRow();

```

```

    int col = purchaseTable.getColumnCount();

```

```

    Object[] data = new Object[col];

```

```

    for (int i=0; i<col; i++)

```

```

        data[i] = purchaseTable.getValueAt(row, i);

```

```

    Integer.parseInt(data[3].toString());

```

```

    data[1].toString();

```

```

}

```

```

private void purchaseButtonActionPerformed(java.awt.event.ActionEvent evt) {

```

```

    productDTO = new ProductDTO();

```

```

    productDTO.setSuppCode("12");

```

```

    productDTO.setProdCode(codeText.getText());

```

```

    try {

```

```

        ResultSet resultSet = new ProductDAO().getProdName(codeText.getText());

```

```

        if (resultSet.next()) {

```

```

            productDTO.setDate("02/01/2020");

```

```

            productDTO.setQuantity(Integer.parseInt(quantityText.getText()));

```

```

            Double costPrice = Double.parseDouble(costText.getText());

```

```

            Double totalCost = costPrice * Integer.parseInt(quantityText.getText());

```

```

            productDTO.setTotalCost(totalCost);

```

```

            new ProductDAO().addPurchaseDAO(productDTO);

```

```

            loadDataSet();

```

```

        } else

```

```

        JOptionPane.showMessageDialog(null, "Please add this product in the \"Products\" section before
proceeding.");

```

```

    } catch (SQLException e) {

```

```

    }

```

```

}

```

```

private void codeTextKeyReleased(java.awt.event.KeyEvent evt) {

```

```

    try {

```

```

        ResultSet resultSet = new ProductDAO().getProdFromCode(codeText.getText());

```

```

        if (resultSet.next()) {

```

```

        nameText.setText(resultSet.getString("productname"));
        costText.setText(String.valueOf(resultSet.getDouble("costprice")));
        sellText.setText(String.valueOf(resultSet.getDouble("sellprice")));
    } else {
        nameText.setText("");
        costText.setText("");
        sellText.setText("");
    }
} catch (SQLException e) {
}
}

```

```

public void loadDataSet() {
    try {
        ProductDAO productDAO = new ProductDAO();
        purchaseTable.setModel(productDAO.buildTableModel(productDAO.getPurchaseInfo()));
    } catch (SQLException throwables) {
    }
}

```

```

public void loadSearchData(String text) {
    try {
        ProductDAO productDAO = new ProductDAO();
        purchaseTable.setModel(productDAO.buildTableModel(productDAO.getPurchaseSearch(text)));
    } catch (SQLException e) {
    }
}

```

```

javax.swing.JTextField codeText;
javax.swing.JTextField costText;
javax.swing.JLabel jLabel1;
javax.swing.JLabel jLabel3;
javax.swing.JLabel jLabel4;
javax.swing.JLabel jLabel6;
javax.swing.JLabel jLabel7;
javax.swing.JLabel jLabel8;
javax.swing.JPanel jPanel1;
javax.swing.JScrollPane jScrollPane1;
javax.swing.JSeparator jSeparator1;
javax.swing.JTextField nameText;
javax.swing.JButton purchaseButton;
javax.swing.JTable purchaseTable;
javax.swing.JTextField quantityText;
javax.swing.JTextField sellText;
}

```

SALES

PURCHASEID	PRODUCTC...	PRODUCTN...	QUANTITY	TOTALCOST
1001	prod1	Laptop	10	850000.0
1002	prod6	Mouse	20	34000.0
1003	prod3	Mobile	5	300000.0
1004	prod5	Charger	5	10000.0
1005	prod2	Laptop	2	140000.0
1006	prod4	Mobile	2	100000.0
1009	prod3	Mobile	2	120000.0
1010	prod7	Power Adapter	10	30000.0
1011	prod8	Smart Watch	20	300000.0

Sales Info

Product Code:

Product Name:

Quantity:

Cost Price:

Selling Price:

Add